1. Extract DB:  
   procedure BrowseEarthquakeData(const Year, Month, Day: Integer; ListView: TListView; ProgramPath: string; MaxMagnitude: Double);

var

FDConnection: TFDConnection;

FDQuery: TFDQuery;

SQLQuery: string;

DateFilter: string;

DatabasePath: string;

DateTimeValue, IntensityAccount: string;

Magnitude, Latitude, Longitude: Double;

Depth: Integer; // Depth as integer

ListItem: TListItem;

DatePart, TimePart: string;

begin

FDConnection := TFDConnection.Create(nil);

FDQuery := TFDQuery.Create(nil);

try

// Construct the database path

DatabasePath := ProgramPath + 'DB\EQCatalogue\_web.sdb';

// Setup SQLite connection

FDConnection.DriverName := 'SQLite';

FDConnection.Params.Database := DatabasePath;

FDConnection.LoginPrompt := False;

FDConnection.Connected := True;

FDQuery.Connection := FDConnection;

// Format date filter

DateFilter := Format('%.4d-%.2d-%.2d', [Year, Month, Day]);

// SQL query to filter data by date, magnitude and extract intensity

SQLQuery := 'SELECT datetime, ms, latitude, longitude, depth, intensity\_account FROM EarthquakeData ' +

'WHERE datetime LIKE :DateFilter AND ms < :MaxMagnitude';

FDQuery.SQL.Text := SQLQuery;

FDQuery.ParamByName('DateFilter').AsString := DateFilter + '%';

FDQuery.ParamByName('MaxMagnitude').AsFloat := MaxMagnitude;

// Execute the query

FDQuery.Open;

// Clear the ListView before populating new data

ListView.Items.Clear;

// Populate the ListView with data

while not FDQuery.Eof do

begin

DateTimeValue := FDQuery.FieldByName('datetime').AsString;

Magnitude := FDQuery.FieldByName('ms').AsFloat;

Latitude := FDQuery.FieldByName('latitude').AsFloat; // Fetch latitude

Longitude := FDQuery.FieldByName('longitude').AsFloat; // Fetch longitude

Depth := FDQuery.FieldByName('depth').AsInteger; // Fetch depth as integer

IntensityAccount := FDQuery.FieldByName('intensity\_account').AsString; // Intensity description

// Split the datetime string into date and time parts

DatePart := Copy(DateTimeValue, 1, 10); // Extract 'YYYY-MM-DD'

TimePart := Copy(DateTimeValue, 12, 8); // Extract 'HH:NN:SS'

// Add a new item to the ListView

ListItem := ListView.Items.Add;

ListItem.Caption := ''; // Leave the checkbox column blank

ListItem.SubItems.Add(DatePart); // Set the date in the second column

ListItem.SubItems.Add(TimePart); // Set the time in the third column

ListItem.SubItems.Add(FormatFloat('0.0', Magnitude)); // Set the magnitude in the fourth column

ListItem.SubItems.Add(FormatFloat('0.000', Latitude)); // Latitude with 3 decimal places

ListItem.SubItems.Add(FormatFloat('0.000', Longitude)); // Longitude with 3 decimal places

ListItem.SubItems.Add(IntToStr(Depth)); // Depth as integer

ListItem.SubItems.Add(IntensityAccount); // Intensity account (description)

// Check the checkbox by default

ListItem.Checked := True; // This line ensures the checkbox is checked

FDQuery.Next;

end;

finally

FDQuery.Free;

FDConnection.Free;

end;

end;

2) EXTRACT DATE TIME:  
procedure ExtractCheckedDateTime(ListView: TListView;

ledEQ\_datex, ledEQ\_timex, ledEQ\_Mag, ledEQ\_Lat, ledEQ\_Lon, ledEQ\_Dep: TLabeledEdit);

var

i: Integer;

DateValue, TimeValue, MagValue, LatValue, LonValue, DepValue: string;

begin

// Iterate over all items in the ListView

// for i := 0 to ListView.Items.Count - 1 do

for i := 7 to ListView.Items.Count - 1 do

begin

// Check if the item is checked (selected)

if ListView.Items[i].Checked then

begin

// Extract Date and Time from SubItems

DateValue := ListView.Items[i].SubItems[0]; // Date is in the first SubItem

TimeValue := ListView.Items[i].SubItems[1]; // Time is in the second SubItem

// Extract Latitude, Longitude, and Depth from SubItems

MagValue := ListView.Items[i].SubItems[2]; // Magnitude in the sixth SubItem

LatValue := ListView.Items[i].SubItems[3]; // Latitude in the third SubItem

LonValue := ListView.Items[i].SubItems[4]; // Longitude in the fourth SubItem

DepValue := ListView.Items[i].SubItems[5]; // Depth in the fifth SubItem

// Set the extracted Date and Time to the labeled edits

ledEQ\_datex.Text := DateValue; // Set the date in ledEQ\_datex

ledEQ\_timex.Text := Copy(TimeValue, 1, 5); // Set the time (HH:MM format) in ledEQ\_timex

// Set the extracted Latitude, Longitude, and Depth to the respective labeled edits

ledEQ\_Mag.Text := MagValue; // Set the magnitude in ledEQ\_Mag

ledEQ\_Lat.Text := LatValue; // Set the latitude in ledEQ\_Lat

ledEQ\_Lon.Text := LonValue; // Set the longitude in ledEQ\_Lon

ledEQ\_Dep.Text := DepValue; // Set the depth in ledEQ\_Dep

// Stop after the first checked item, if you only need one

Break;

end;

end;

end;

3) set datetimepicker = date /time of the eq. event:  
procedure SetDateTimePickerFromLabeledEdits(ledEQ\_datex, ledEQ\_timex: TLabeledEdit; dtpEQDateTime: TDateTimePicker);

var

CombinedDateTime: TDateTime;

DateStr, TimeStr: string;

DateValue, TimeValue: TDateTime;

FormatSettings: TFormatSettings;

begin

// Set up custom format settings

FormatSettings := TFormatSettings.Create;

FormatSettings.DateSeparator := '-';

FormatSettings.ShortDateFormat := 'yyyy-mm-dd';

FormatSettings.TimeSeparator := ':';

FormatSettings.ShortTimeFormat := 'hh:nn';

// Get values from labeled edits

DateStr := Trim(ledEQ\_datex.Text); // Example: '2024-10-15'

TimeStr := Trim(ledEQ\_timex.Text); // Example: '14:30'

try

// Convert date and time strings to TDateTime

DateValue := StrToDate(DateStr, FormatSettings);

TimeValue := StrToTime(TimeStr, FormatSettings);

// Combine the date and time

CombinedDateTime := DateValue + Frac(TimeValue);

// Set the value of the DateTimePicker

dtpEQDateTime.DateTime := CombinedDateTime;

except

on E: Exception do

ShowMessage('Error setting DateTimePicker: ' + E.Message);

end;

end;